CONOSCERE
IL COMPUTER DIRETTAMENTE DAL COMPUTER

per Commodore Vic20 e 64

Beatrice d'Este

Nella lezione 21
ti ho spiegato il
funzionamento dei
files sequenziali su
nastro per memorizzare
sulla cassetta i dati
di un programma.
I files sequenziali possono
essere usati nello stesso
modo anche su disco, l'unica
differenza è nell'istruzione OPEN,
che dovrà aprire l'accesso al disco
anzichè al registratore.

Per aprire su disco un file sequenziale per la registrazione dei dati, dovrai scrivere: **OPEN1,8,2,"DATI,S,W".**
Invece per caricare i dati: **OPEN1,8,2,"DATI,S,R".**
Sul disco, oltre che i files sequenziali, possono essere usati anche quelli casuali: RANDOM e RELATIVI. I RANDOM servono per memorizzare dati in qualsiasi posizione del disco, specificandone la traccia e il settore.
Il loro uso comunque è molto raro.
I RELATIVI invece hanno un impiego molto più vasto soprattutto per le applicazioni **professionali**; quando si intende trattare una grande quantità di dati che devono essere rintracciati velocemente.
Infatti, mentre in un file sequenziale per leggere un determinato record è necessario prima leggere tutti i record precedenti; nei files relativi è possibile leggere direttamente il record voluto specificandone il numero.

Naturalmente un file relativo prima di essere usato dovrà essere creato, usando l'istruzione OPEN come nel seguente esempio:
OPEN2,8,2,"NOME FILE,L,"+CHR$(1ØØ).
Prima di aprire un file relativo è consigliabile aprire il canale dei comandi al disco con l'indirizzo secondario 15, quindi:
OPEN15,8,15.
Questo servirà poi per vedere, nel corso delle operazioni, se si presentano errori.
Usando infatti l'istruzione **INPUT#15,A,B$,C,D** verranno assegnati i codici di errore alle variabili nel seguente modo:

A - numero dell'errore (se è minore di 20 non si tratta di un errore)
B$ - descrizione dell'errore
C - traccia in cui si è verificato l'errore
D - settore in cui si è verificato l'errore

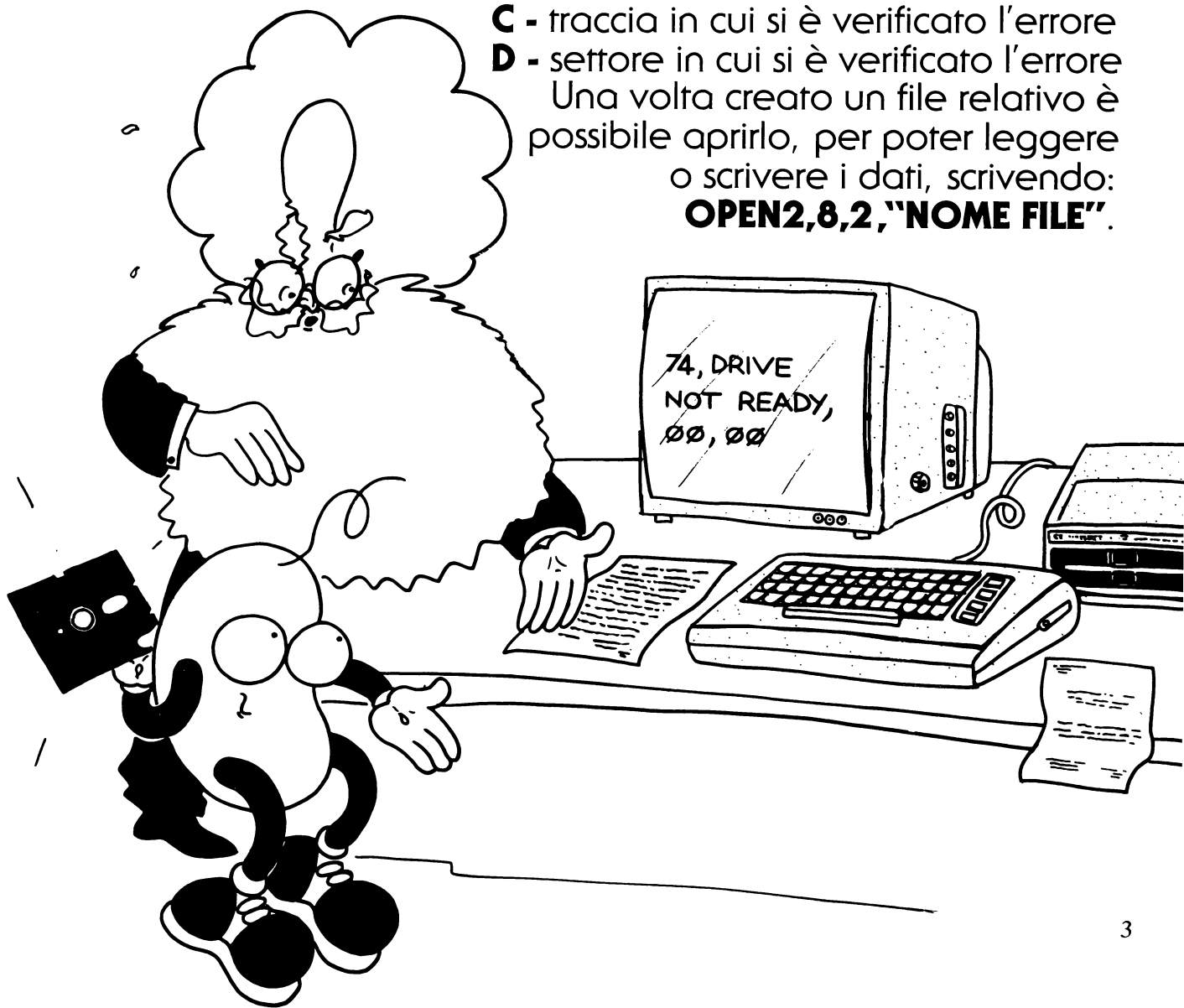
Una volta creato un file relativo è possibile aprirlo, per poter leggere o scrivere i dati, scrivendo:
OPEN2,8,2,"NOME FILE".

Prima di compiere operazioni di lettura o di scrittura di un record è necessario posizionare il PUNTATORE AL RECORD su quello desiderato, con la seguente istruzione:
PRINT#15, "P"CHR$ (2) CHR$ (BL) CHR$ (BH) CHR$ (P).
Per scrivere i dati nel file, dopo essersi posizionati sul record desiderato, basterà usare l'istruzione PRINT .
Ad esempio: **PRINT# 2,X$.**
Per leggere invece i dati in un file relativo, sempre dopo aver posizionato il puntatore al record voluto, è possibile usando l'istruzione INPUT#, oppure GET# se si vuole leggere un carattere per volta.
Ad esempio: **INPUT#2,X$.**

TABELLA ELENCO DEI MESSAGGI DI ERRORE DELL'UNITA' A FLOPPY

0	Non vi sono errori
1	Messaggio in risposta alla cancellazione di file. Non vi sono errori.
2-19	Non usati come messaggio errore. Da ignorare.
20	Intestazione del blocco non trovato sul disco.
21	Mark di sincronismo non trovato.
22	Blocco dati non presente.
23	Errore checksum nei dati
24	Errore di codifica del byte.
25	Errore di verifica in scrittura.
26	Tentativo di scrittura con protezione inserita.
27	Errore di checksum nell'intestazione.
28	Estensione dei dati nel blocco successivo.
29	Mancata coincidenza dell'ID del disco.
30	Errore generico di sintassi.
31	Comando non valido.
32	Linea troppo lunga.
33	Nome di file non valido.
34	Nessun file è dato.
39	File comando non trovato.
50	Record non presente.
51	Overflow su record.
52	File troppo grande.
60	File aperto per scrittura.
61	File non aperto.
62	File non trovato.
63	File già esistente.
64	Verifica fallita per il tipo di file.
65	Nessun blocco.
66	Traccia o settore illegale.
67	Traccia o settore illegale del sistema.
70	Nessun canale disponibile.
71	Errore nel direttorio.
72	Disco o direttorio completo.
73	Messaggio di accensione o tentativo di scrittura con versione DOS non compatibile.
74	Unità a disco non pronta.

Listato dell'esercizio: ESEMPIO PRATICO (CBM 64)

```
10 poke53280,0:poke53281,0:i=920
20 input"parola:";p$
30 if(p$="")+(len(p$)>12)then20
40 print"":v=int(i/(len(p$)+1))
50 fork=1tov
60 c$=mid$("",int(rnd(0)*7+1),1)
70 printc$;p$"  ";
80 nextk
90 fort=1to4000:nextt:print""
```

Listato dell'esercizio: ESEMPIO PRATICO (VIC 20)

Il listato è identico a quello per il CBM 64, tranne nella linea:

```
10 poke36879,8:i=484
```

Listato dell'esercizio: COMPLETA IL LISTATO

```
10 forx=1to5:fory=1to[*]
20 l$(x)=[*]+chr$(int(rnd(0)*5+71))
30 nexty
40 printx;l$(x):nextx:s=[*]
50 forx=1to4:[*]y=1to4
60 ifl$([*])<l$(y+1)then80
70 h$=l$(y):l$(y)=l$(y+1):l$(y+1)=[*]
80 nexty:nextx
90 for[*]to5
100 print"scrivi la stringa";x
110 inputa$:ifa$<>l$(x)then[*]
120 [*]x
130 f=ti:print"tempo:";int((f-s)/[*])
140 goto[*]
150 print"risposta: ";[*](x)
160 end
```

PROGRAMMIAMO INSIEME (CBM 64)

```
10 poke53280,7:poke53281,7
15 open15,8,15:k$=chr$(13)
20 print"1 - creazione file"
30 print"2 - inserimento"
40 print"3 - ricerca"
50 print"4 - fine programma"
60 input"scelta (1-4)";s
70 if(s<1)+(s>4)then60
80 ongoto100,200,600,900
100 p=100
120 print"attendi un attimo"
140 open2,8,2,"dati,l,"+chr$(74)
150 gosub1000
160 print#15,"p"chr$(2)chr$(l)chr$(h)
170 print#2,"*":gosub2000
180 close2:goto20
200 input"inserimento codice";i
220 if(i<1)+(i>100)then200
300 input"nome:";n$
310 if(len(n$)=0)+(len(n$)>20)then300
320 input"indirizzo:";i$
330 if(len(i$)=0)+(len(i$)>20)then320
340 input"citta':";c$
350 if(len(c$)=0)+(len(c$)>20)then340
360 input"telefono:";t$
370 if(len(t$)=0)+(len(t$)>10)then360
380 open2,8,2,"dati"
390 p=i:gosub1000
400 print#15,"p"chr$(2)chr$(l)chr$(h)chr$(1)
420 print#2,n$k$i$k$c$k$t$:gosub2000
440 close2:goto20
600 input"ricerca codice";r
610 if(r<1)+(r>100)then600
620 p=r:gosub1000
630 open2,8,2,"dati"
640 print#15,"p"chr$(2)chr$(l)chr$(h)chr$(1)
660 gosub2000:input#2,n$,i$,c$,t$
670 close2
680 printn$:printi$:printc$:printt$
700 gosub3000:goto20
900 close15:end
1000 h=int(p/256):l=p-h*256:return
2000 input#15,a,b$,c,d
2010 if(a<20)+(a=50)thenreturn
2020 printa;b$;c;d:end
3000 gety$:ify$=""then3000
3020 return
```

PROGRAMMIAMO INSIEME (VIC 20)

Il listato è identico a quello per il CBM 64, tranne nella linea:

```
10 poke36879,127
```

Soluzione dell'esercizio: COMPLETA IL LISTATO CBM 64
(lez. 28)

```
10 poke53280,2:poke53281,2
20 printchr$(147):v=1024:c=55296
30 forx=0to19
40 p=x*40+19-x
50 pokev+p,233:pokec+p,7
60 ifx=0theny=20:goto110
70 fory=20-xto19+x
80 p=x*40+y
90 pokev+p,160:pokec+p,7
100 nexty
110 p=x*40+y:pokev+p,223:pokec+p,7
120 nextx
130 forv=1to5:fors=1to6
140 poke53280,s:poke53281,s
150 fork=1to200:nextk
160 nexts,v:end
```

Soluzione dell'esercizio: COMPLETA IL LISTATO VIC 20
(lez. 28)

```
10 poke36879,42
20 printchr$(147):v=7680:c=38400
30 forx=0to10
40 p=x*22+10-x
50 pokev+p,233:pokec+p,7
60 ifx=0theny=11:goto110
70 fory=11-xto10+x
80 p=x*22+y
90 pokev+p,160:pokec+p,7
100 nexty
110 p=x*22+y:pokev+p,223:pokec+p,7
120 nextx
130 forv=1to5:fors=1to6
140 poke36879,s*17+8
150 fork=1to200:nextk
160 nexts,v:end
```